REMARKS

Claims 1-15 stand rejected. By this Amendment, claim 1 has been amended, claims 14 and 15 have been canceled and new claims 16-20 have been added to the application. Claims 1-13 and 16-20 are therefore pending. It is believed that each of the pending claims define an invention which is novel and unobvious over the cited art. Favorable reconsideration of this case is respectfully requested.

The specification has been reviewed and edited to eliminate minor inaccuracies and typographical errors.

The present invention provides a method and system for identifying and restricting operation of an unauthorized software program. In a preferred embodiemt, a key resides in a first non-volatile part of a computer's memory. The non-volatile memory being typically, but not necessarily, a stand alone module which is not erasable and therefore cannot be modified (see the present specification, page 9, lines 3 to 7). A verification structure is formed to include one or more license records, described below, and resides in a second non-volatile part of the memory, (see the present specification, page 9, lines 8 to 10). The second non-volatile part is erasable and therefore license data in the verification structure can be modified. For example, license data may be added or modified as required, for example, when new licenses are added or expire. The license records are obtained by encrypting license records extracted from the software program with the key stored in the first non-volatile part of the computer's memory, page 9 lines 19 to 21. The key may be of many possible variants (see, for example, the options elaborated in the bridging paragraph between pages 6 and 7 of the specification). The key may also be used for encryption of license record or decryption of encrypted license record all as required and appropriate (see, e.g. page 7 lines 20, 21). Moreover, the contents of the license record is very flexible (see e.g. page 10 lines 17 to 25). The specification explains other advantages of the

invention in more detail.

Claims 1-4, 6 and 10-13 have been rejected under 35 U.S.C. 102(e) as being unpatentable over U.S. Patent No. 5,892,900 to Ginter et al.

Ginter et al. do not anticipate the present invention as they do not disclose, among other things, setting up a verification structure and verifying the program using the verification structure as recited in the rejected claims.

Ginter et al. provide a system and method for secure electronic transaction management and electronic rights protection. Ginter's method provides "machine bound" delivery of content or software through what they call "Stationary Object" (col. 136, lines 64-66 and Fig 18). A stationary object is an object bound to a specific machine. The main security measure used to protect the content of a "Stationary Object" from illegal use is to encrypt it according to the target's unique key (col. 137, lines 45-50).

"For example, a container that is bound by its control to a specific VDE node is called a "stationary Object (see Fig 18)" (col. 136, lines 64-66). "Fig 18 shows an example of a "stationary object" structure 850 provided by the preferred embodiment. 'Stationary Object" structure is intended to be used only at specific VDE electronic appliance/installations that have received explicit permissions to use one or more portions of the stationary object..." (col. 137, lines 23-28)

"This private body (method) section 806 is preferably encrypted using one or more private body keys contained in the separate permissions record 808. The data blocks 812 contain content (information or administrative) that may be encrypted using one or more content keys also provided in permissions record 808."

Accordingly, in Ginter et al., software distributed through a stationary object is encrypted for the specific machine therefor "bound" to it. " Objects may be classified in one sense based on

whether the protection information is bound together with the protected information" (Ginter, col. 136, line 62).

Consequently, this method suffers from the deficiency that it is incompatible with free "out of channel" or "retail channel" distribution. In the latter mode of operation, it is often desired to broadcast a single version of the software to all the subscribers, rather than a machine bound (and obviously different) version for each subscriber that is required by Ginter et al. In other words, the "Stationary Object" aspect of Ginter has the shortcoming, among others, that it cannot support a business model where the distributor doesn't know the final target machine. Therefore, the system and method will not be able to freely distribute the software, such as happens in retail and software companies that ships millions of copies.

Ginter itself acknowledges that the problem with "Stationary Objects" therefore suggests a second method named "Traveling Objects" (col. 136, line 66 - col.137, line 3, and fig. 19). A "Traveling Object" is an object that contains the information needed to use its content:
"a container that is not bound by its control information to a specific VDE node but rather carries sufficient control and permissions to permit its use, in a whole or in part, at any of several sites is called a "Traveling Object" (Ginter, col. 136, line 66 - col. 137, line 3). A traveling object allows shipping the content to unknown destinations by encrypting the content with the same key again and again. However, Ginter uses an encryption technique in the "Traveling Object" feature in which the key is incorporated in the distributed objects. Ginter acknowledge the shortcomings of this solution to wit:

> "In the case of a "traveling object", content owners may distribute information with some or all of the key blocks **810** included in the object **300** in which the content is encapsulated. Putting keys in distributed objects **300** increases the exposure to attempts to defeat security mechanisms by breaking or cryptoanalyzing the encryption algorithm with which the private header is protected (e.g., by determining the key for the header's encryption). This breaking of security would normally require considerable skill and time, but if broken, the

> algorithm and key could be published so as to allow large numbers of individuals who possess objects that are protected with the same key(s) and algorithm(s) to illegally use protected information. (Col. 139, lines 38 to 50)."

Ginter admits that this solution can thus be used only with limited type of software which is not commercially valuable, to wit:

> "As a result, placing keys in distributed objects **300** may be limited to content that is either "time sensitive" (has reduced value after the passage of a certain period of time), or which is somewhat limited in value, or where the commercial value of placing keys in objects (for example convenience to end-users, lower cost of eliminating the communication or other means for delivering keys and/or permissions information and/or the ability to supporting objects going "out-of channel") exceeds the cost of vulnerability to sophisticated hackers. (Col. 139, lines 50 to 59)."

The present invention differs from and overcomes the deficiencies associated with the stationary object and traveling object methods described in Ginter et al. In the present invention, a unique key is stored in the first non-volatile memory of the computer. A software program in the volatile memory of the computer is selected. A license record is extracted from the software program and encrypted using the unique key stored in the computer (see new independent claim 20). Thus, the software program is not machine bound as is required by the stationery object method, nor is the same key used over and over to encrypt the software as is the case with the traveling object. In the present method, the verification structure is formed by using a unique key for each computer and license record information in the software.

Moreover, in col. 70, line 23 – col. 71, line 25 Ginter et al. describe the architecture as add-on hardware which is named "SPU"(col. 63, line 66 – col. 64, line 15). Col. 64, lines 16-21 explicitly detail the fact that the SPU is a hardware add-on, not part of the PC. In col. 70 Ginter et al. describes the memory architecture for the SPU and uses terms taken from the PC engineering world. However, this is not referring to those actual PC components which name is used in their design.

In view of the above, it is clear that Ginter et al do not describe the step of setting up a verification structure. The portions of Ginter et al. referred to by the Examiner all describe the elements of the proprietary hardware of Ginter et al. These portions of Ginter et al. do not describe setting the verification structure in memory, they describe basic functionality of a common CPU that loads code to memory and executes it.

Furthermore, it is clear that Ginter et al. do not describe the step of verifying the program using the verification structure. There is no mention whatsoever in Ginter et al. in col. 70, lines 23-53 and col. 63, line 67 - col. 64, line 15 referred to by the Examiner of a process where a software program verifies its authenticity using a license (verification structure) stored in the second volatile non-volatile memory. The functionality described in these portions of Ginter et al. is the different functionality that add-on hardware, referred to as SPU, can perform. There no specific discussion as to how the functionality is performed and whether it is actually has something to do with protecting software.

In contrast to Ginter et al., the present invention provides a system and method which not only enables free distribution of the software (such as happens in retail stores, and software companies that ship millions of copies), that overcomes the problems with the stationary object in Ginter et al., but also does not suffer from the limitations of incorporating the key in the distributed data as is the case with the traveling object of Ginter et al. Moreover, the steps of setting up a verification structure and using that structure for verification are clearly recited in the rejected claims

For example, independent claim 1 recites a method of restricting software operation within a license limitation. The method is useful for a computer including a first, non-erasable, non-volatile memory area, a second, erasable, non-volatile memory area, and a volatile memory area. The first non-volatile memory accommodates data that includes unique key. According to

the method of the invention, a program residing in the volatile memory is selected. A verification structure is set up in the second non-volatile memory. The verification structure accommodates data that include at least one license record. The program is verified using at least the verification structure. Based on the verification, the program is acted on accordingly.

Additionally, new independent claim 20 recites additional features not disclosed in Ginter et al. In claim 20, a method for restricting access to a software program is defined. The method includes storing a pseudo-unique key in a first non-volatile memory area of a computer. A software program residing in a volatile memory area of the computer is selected. License information is extracted from the software program. The license information is encrypted using the pseudo-unique key. The encrypted pseudo-unique key is stored in a second non-volatile memory area of the computer. The software program is verified using based on the encrypted pseudo-unique key and the software program is acted on based on the verification.

Thus, in the method recited in claim 20, license information is extracted from the software program and encrypted using a key stored on the computer. Applicants review of the cited references did not reveal any description of extracting information from a program, encrypting the information using a key stored on the computer, and storing the encrypted information on the computer. There is no description in the cited references of the steps of "extracting license information from the software program" and "encrypting the license information using the pseudo-unique key" as is recited in new claim 22.

No claim recitation can be ignored in determining anticipation. See Pac-Tex, Inc. v. Amerace Corp., 14 U.S.P.Q.2d 187, (Fed. Cir. 1990). Anticipation requires the disclosure, in a prior art reference, of each and every recitation as set forth in the claims. See Titanium Metals Corp. v. Banner, 227 U.S.P.Q. 773 (Fed. Cir. 1985), Orthokinetics, Inc. v. Safety Travel Chairs,

Inc. 1 U.S.P.Q.2d 1081 (Fed. Cir. 1986), and Akzo N.V. v. U.S. International Trade Commissioner, 1 U.S.P.Q.2d 1241 (Fed. Cir. 1986).

There must be no difference between the claimed invention and reference disclosure for an anticipation rejection under 35 U.S.C. 102. See Scripps Clinic and Research Foundation v. Genentech, Inc., 18 U.S.P.Q.2d 1001 (CAFC, 1991) and Studiengesellschaft Kohle GmbH v. Dart Industries, 220 U.S.P.Q. 841 (CAFC, 1984).

In view of the above discussion, it is clear that the cited reference does not teach each and every element recited in the claims as required by 35 U.S.C. 102(e). Therefore, the withdrawal of the rejection of claims 1-4, 6 and 10-14 under 35 U.S.C. 102(e) is respectfully requested.

Claims 5 and 7-9 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Ginter et al. in view of Goldman et al.

Claims 5 and 7-9 depend from independent claim 1 and would patentable for at least the reasons discussed above regarding independent claim 1.

Goldman et al. do not supplement Ginter et al. to teach or suggest the features as recited in the rejected claims.

Claims 14 and 15 have been rejected under 35 U.S.C. 103(a) as being unpatentable over Ginter et al in view of Goldman et al.

Claims 14 and 15 have been canceled, rendering this rejection moot.

In view of the above discussion, it is clear that the cited references, taken alone of in combination, do not render the present invention obvious. Therefore the withdraw of this rejection is respectfully requested.

Attached hereto is a marked-up version of the changes made to the specification and claims by the current amendment. The attached page is captioned "**Version with markings to show changes made.**"

In view of the foregoing, reconsideration and allowance of this application are believed in order, and such action is earnestly solicited.

The Commissioner is authorized to charge any fee necessitated by this Amendment to our Deposit Account No. 22-0261.

Respectfully submitted,

VENABLE, Attorneys at Law

Robert Kinberg
Registration No. 26,924
P.O. Box 34385
Washington, D.C. 20043-9998
Telephone 202-962-4800
Telefax 202-962-8300

RK/JAK/lrh
#289169

VERSION WITH MARKINGS TO SHOW CHANGES MADE

IN THE SPECIFICATION

Page 1, please rewrite paragraph 2 as follows:

Numerous methods have been devised for the identifying and restricting of an unauthorized software program's operation. These methods have been primarily motivated by the grand proliferation of illegally copied software, which is engulfing the marketplace. This illegal copying represents billions of dollars in lost profits to commercial software developers.

Page1, please rewrite paragraph 3 as follows:

Hardware based products have also been developed to validate authorized software usage by accessing a dongle that is coupled e.g. to the parallel port of the P.C. These units are expensive, inconvenient, and not particularly suitable for software that may be sold by downloading (e.g. over the internet).

Page 9, please rewrite paragraph 3 as follows:

The second non-volatile memory includes a license-record-area (9) e.g. ~~for the containing of~~ which contains at least one encrypted license-record (e.g. three records 10-12). The volatile memory accommodates a license program (16) having license record fields (13-15) appended thereto. By way of example said fields stand for Application names (e.g. Lotus 123), Vendor name (Lotus inc.), and ~~no~~ number of licensed copies (1 for stand alone usage, >1 for number of licensed users for a network application).

Page 9, please rewrite paragraph 4 as follows:

Those versed in the art will readily appreciate that the license record is not necessarily bound to ~~continuos~~ continuous fields. In fact, the various license content components of the data

record may be embedded in various locations in the application. Any component may, if desired, be encrypted.

Page 9 and continuing on page 10, please rewrite paragraph 7 as follows:

The bureau forms the proposed license-record from the contents, encrypts (utilizing predetermined encryption algorithm) the so formed license-record using the key (8), and compares the so formed encrypted license-record with the license-record (10-12). The bureau generates an overlay according to the result of the comparison ~~indication~~ indicating successful comparison, non-critical failure comparison and the critical failure comparison.

IN THE CLAIMS:

Please amended the claims as follows:

1. (Amended) A method of restricting software operation within a license ~~limitation comprising;~~ for use with a computer including ~~having~~ a first, non erasable, non-volatile memory area, a second, non-erasable non-volatile memory area, and a volatile memory area; the first non volatile memory accomodates data that includes unique key; the method comprising the steps of:

selecting a program residing in the volatile memory,

setting up a verification structure in the second non-volatile memory ~~memories~~, the verfication structure accommodates data that includes at least one license record,

verifying the program using at least said verification structure~~the structure~~, and

acting on the program according to the verification.

Please add the following new claims:

16. (New) The method according to Claim 1, wherein the unique key includes a pseudo-unique key.

17. (New) The method according to Claim 1, wherein said step of setting up a verification record, including the license record, includes encrypting a license record data in said program using at least said key.

18. (New) The method according to Claim 1, wherein said step of verifying the program includes decrypting the license record data accommodated in said second non volatile memory using at least said unique key.

19. (New) The method according to Claim 1, wherein said step of verifying the program includes encrypting the license record that is accommodated in said program using at least said unique key.

20. (New) A method for restricting access to a software program, comprising:

storing a pseudo-unique key in a first non-volatile memory area of a computer;

selecting a software program residing in a volatile memory area of the computer;

extracting license information from the software program;

encrypting the license information using the pseudo-unique key;

storing the encrypted pseudo-unique key in a second non-volatile memory area of the computer;

verifying the software program using based on the encrypted pseudo-unique key; and

acting on the software program based on the verification.